Web Services Policy Framework (WS-Policy)

September 2004

Authors

Siddharth Bajaj, VeriSign
Don Box, Microsoft
Dave Chappell, Sonic Software
Francisco Curbera, IBM
Glen Daniels, Sonic Software
Phillip Hallam-Baker, VeriSign
Maryann Hondo, IBM
Chris Kaler, Microsoft
Dave Langworthy, Microsoft
Ashok Malhotra, Microsoft
Anthony Nadalin, IBM
Nataraj Nagaratnam, IBM
Mark Nottingham, BEA
Hemma Prafullchandra, VeriSign
Claus von Riegen, SAP
Jeffrey Schlimmer (Editor), Microsoft
Chris Sharp, IBM
John Shewchuk, Microsoft

Copyright Notice

Abstract

The Web Services Policy Framework (WS-Policy) provides a general purpose model and corresponding syntax to describe the policies of a Web Service.

WS-Policy defines a base set of constructs that can be used and extended by other Web services specifications to describe a broad range of service requirements and capabilities.

Composable Architecture

The Web service specifications (WS*) are designed to be composed with each other to provide a rich set of tools for secure, reliable, and/or transacted Web services. WS-Policy by itself does not provide a negotiation solution for Web services. WS-Policy is a building block that is used in conjunction with other Web service and application-specific protocols to accommodate a wide variety of policy exchange models.

Status

This WS-Policy Specification is a public draft release and is provided for review and evaluation only. The Authors hope to solicit your contributions and suggestions in the near future. The Authors make no warrantees or representations regarding the specifications in any manner whatsoever.

Table of Contents

1. Introduction

WS-Policy provides a flexible and extensible grammar for expressing the capabilities, requirements, and general characteristics of entities in an XML Web services-based system. WS-Policy defines a framework and a model for the expression of these properties as policies.

WS-Policy defines a policy to be a collection of policy alternatives, where each policy alternative is a collection of policy assertions. Some policy assertions specify traditional requirements and capabilities that will ultimately manifest on the wire (e.g., authentication scheme, transport protocol selection). Other policy assertions have no wire manifestation yet are critical to proper service selection and usage (e.g., privacy policy, QoS characteristics). WS-Policy provides a single policy grammar to allow both kinds of assertions to be reasoned about in a consistent manner.

WS-Policy does not specify how policies are discovered or attached to a Web service. Other specifications are free to define technology-specific mechanisms for associating policy with various entities and resources. WS-PolicyAttachment [WS-PolicyAttachment] defines such mechanisms, especially for associating policy with arbitrary XML elements, WSDL artifacts, and UDDI elements. Subsequent specifications will provide profiles on WS-Policy usage within other common Web service technologies.

1.1 Goals

The goal of WS-Policy is to provide the mechanisms needed to enable Web services applications to specify policy information. Specifically, this specification defines the following:

- An XML Infoset called a *policy expression* that contains domain-specific, Web Service policy information.
- A core set of constructs to indicate how choices and/or combinations of domain-specific policy assertions apply in a Web services environment.

WS-Policy is designed to work with the general Web services framework, including WSDL service descriptions [WSDL] and UDDI service registrations [UDDIAPI20, UDDIDataStructure20, UDDI30].

1.2 Example

The following example illustrates a policy:

```
01 <wsp:Policy>
```

```
02   <wsp:ExactlyOne>
03     <wsse:SecurityToken>
04       <wsse:TokenType>wsse:Kerberosv5TGT</wsse:TokenType>
05     </wsse:SecurityToken>
06     <wsse:SecurityToken>
07       <wsse:TokenType>wsse:X509v3</wsse:TokenType>
08     </wsse:SecurityToken>
09   </wsp:ExactlyOne>
10 </wsp:Policy>
```

This example illustrates a security policy using assertions defined in WS-SecurityPolicy [WS-SecurityPolicy]. Lines 01-10 represent a policy for authentication.

Editor's Note: We expect the assertions defined in WS-Security Policy to be updated, but the current assertions are used herein for the time being.

Lines 02-09 illustrate the Exactly One policy operator. Policy operators group policy assertions into policy alternatives. A valid interpretation of the policy above would be that an invocation of a Web service contains one of the security token assertions (Lines 03-08) specified.

Lines 03-05 and 06-08 represent two specific security policy assertions that indicate that two types of authentication are supported.

2. Notations and Terminology

This section specifies the notations, namespaces, and terminology used in this specification.

2.1 Notational Conventions

The keywords "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC 2119].

Namespace [XML-NS] URIs represent application-dependent or context-dependent URIs as defined in RFC 2396 [RFC 2396].

This specification uses the following syntax within normative outlines:

- The syntax appears as an XML instance, but values in *italics* indicate data types instead of values.
- Characters are appended to elements and attributes to indicate cardinality:
  - "?" (0 or 1)
  - "*" (0 or more)
  - "+" (1 or more)
- The character "|" is used to indicate a choice between alternatives.
- The characters "[" and "]" are used to indicate that contained items are to be treated as a group with respect to cardinality or choice.

Elsewhere in this specification, the characters "[" and "]" are used to call out references and XML Infoset property names.

2.2 Extensibility

Within normative outlines, ellipses (i.e., "...") indicate a point of extensibility that allows other Element or Attribute Information Items. Information Items MAY be added at the indicated extension points but MUST NOT contradict the semantics of the Element Information Item indicated by the **[parent]** or **[owner]** property of the extension. If a processor does not recognize an Attribute Information Item, the processor SHOULD ignore it; if a processor does not recognize an Element Information Item, the processor SHOULD treat it as an assertion.

2.3 Namespaces

The XML namespace URI that MUST be used by implementations of this specification is:

```
http://schemas.xmlsoap.org/ws/2004/09/policy
```

A normative copy of the XML Schema [XMLSchema1] for WS-Policy constructs may be retrieved by resolving the URI "http://schemas.xmlsoap.org/ws/2004/09/policy".

The following namespaces are used in this document:

Prefix	Namespace
wsdl	http://schemas.xmlsoap.org/wsdl/
wsse	http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd
wsp	http://schemas.xmlsoap.org/ws/2004/09/policy
wsu	http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd
xs	http://www.w3.org/2001/XMLSchema

In this document reference is made to the `@wsu:Id` attribute in a utility schema (http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd). The `@wsu:Id` attribute is included in the utility schema with the intent that other specifications requiring a global Id could reference it (as is done here).

2.4 Terminology

We introduce the following terms that are used throughout this document:

Policy – A *policy* is a collection of policy alternatives.

Policy Alternative – A *policy alternative* is a collection of policy assertions.

Policy Assertion – A *policy assertion* represents an individual requirement, capability, or other property of a behavior.

Policy Assertion Type – A *policy assertion type* represents a class of policy assertions and implies a schema for instances of the assertion and assertion-specific semantics.

Policy Vocabulary – The *policy vocabulary* of a policy is the set of all policy assertion types used in the policy.

Policy Expression – A *policy expression* is an XML Infoset representation of a policy, either in a normal form or in an equivalent compact form.

Policy Subject – A *policy subject* is an entity (e.g., an endpoint, message, resource, interaction) with which a policy can be associated.

Policy Scope – A *policy scope* is a collection of policy subjects to which a policy may apply.

Policy Attachment – A *policy attachment* is a mechanism for associating policy with one or more policy scopes.

3. Policy Model

This section defines an abstract model for policies and for operations upon policies.

This abstract model is independent of how it is represented as an XML Infoset.

3.1 Policy Assertion

A policy assertion identifies a behavior that is a requirement (or capability) of a policy subject. Assertions indicate domain-specific (e.g., security, transactions) semantics and are expected to be defined in separate, domain-specific specifications.

Assertions are strongly typed. The type is identified only by the XML Infoset **[namespace name]** and **[local name]** properties (that is, the qualified name or QName) of the root Element Information Item representing the assertion. Assertions of a given type MUST be consistently interpreted independent of their policy subjects.

The XML Infoset of an assertion MAY contain a non-empty **[attributes]** property and/or a non-empty **[children]** property. Such content MAY be used to parameterize the behavior indicated by the assertion. For example, an assertion identifying support for a specific reliable messaging mechanism might include an Attribute Information Item to indicate how long an endpoint will wait before sending an acknowledgement. However, additional assertion content is not required when the identity of the root Element Information Item alone is enough to convey the requirement (capability).

3.2 Policy Alternative

A policy alternative is a potentially empty collection of policy assertions. An alternative with zero assertions indicates no behaviors. An alternative with one or more assertions indicates behaviors implied by those, and only those assertions.

The vocabulary of a policy alternative is the set of all assertion types within the alternative. The vocabulary of a policy is the set of all assertion types used in the policy. An assertion whose type is part of the policy's vocabulary but is not included in an alternative is explicitly prohibited by the alternative.

Assertions within an alternative are not ordered, and thus aspects such as the order in which behaviors (indicated by assertions) are applied to a subject are beyond the scope of this specification.

A policy alternative MAY contain multiple instances of an assertion type. Mechanisms for determining the aggregate behavior indicated by the assertion instances (and their Post-Schema-Validation Infoset (PSVI) content, if any) are specific to the assertion type and are outside the scope of this document.

3.3 Policy

At the abstract level a policy is a potentially empty collection of policy alternatives. A policy with zero alternatives contains no choices; a policy with one or more alternatives indicates choice in requirements (or capabilities) within the policy.

Alternatives are not ordered, and thus aspects such as preferences between alternatives in a given context are beyond the scope of this specification.

Alternatives within a policy may differ significantly in terms of the behaviors they indicate. Conversely, alternatives within a policy may be very similar. In either case, the value or suitability of an alternative is generally a function of the semantics of assertions within the alternative and is therefore beyond the scope of this specification.

3.4 Web services

Applied in the Web services model, policy is used to convey conditions on an interaction between two Web service endpoints. Satisfying assertions in the policy usually results in behavior that reflects these conditions. Typically, the provider of a Web service exposes a policy to convey conditions under which it provides the service. A requester might use this policy to decide whether or not to use the service. A requester may choose any alternative since each is a valid configuration for interaction with the service, but a requester must choose only a single alternative since each is an alternative configuration.

A *policy assertion* is *supported* by a requester if and only if the requester satisfies the requirement (or accommodates the capability) corresponding to the assertion. A *policy alternative* is *supported* by a requester if and only if the requester supports all the assertions in the alternative. And, a *policy* is *supported* by a requester if and only if the requester supports at least one of the alternatives in the policy. Note that although policy alternatives are meant to be mutually exclusive, it cannot be decided in general whether or not more than one alternative can be supported at the same time.

Note that a requester may be able to support a policy even if the requester does not understand the type of each assertion in the vocabulary of the policy; the requester only has to understand the type of each assertion in the vocabulary of a *policy alternative*. This characteristic is crucial to versioning and incremental deployment of new assertions because this allows a provider's policy to include new assertions in new alternatives while allowing requesters to continue to use old alternatives in a backward-compatible manner.

4. Policy Expression

To convey policy in an interoperable form, a policy expression is an XML Infoset representation of a policy. The normal form policy expression is the most straightforward Infoset; equivalent, alternative Infosets allow compactly expressing a policy through a number of constructs.

4.1 Normal Form Policy Expression

To facilitate interoperability, this specification defines a normal form for policy expressions that is a straightforward XML Infoset representation of a policy, enumerating each of its alternatives that in turn enumerate each of their assertions. The schema outline for the normal form of a policy expression is as follows:

```
<wsp:Policy ... >
  <wsp:ExactlyOne>
    [ <wsp:All> [ <Assertion ...> ... </Assertion> ]* </wsp:All> ]*
  </wsp:ExactlyOne>
</wsp:Policy>
```

The following describes the Element Information Items defined in the schema outline above:

/wsp:Policy

A policy expression.

/wsp:Policy/wsp:ExactlyOne

A collection of policy alternatives. If there are no Element Information Items in the **[children]** property, there are no admissible policy alternatives, i.e., no behavior is admissible.

/wsp:Policy/wsp:ExactlyOne/wsp:All

A policy alternative; a collection of policy assertions. If there are no Element Information Items in the **[children]** property, this is an admissible policy alternative that is empty, i.e., no behavior is specified.

*/wsp:Policy/wsp:ExactlyOne/wsp:All/**

XML Infoset representation of a policy assertion.

To simplify processing and improve interoperability, the normal form of a policy expression should be used where practical.

For example, the following is the normal form of the policy expression example introduced earlier.

```
01 <wsp:Policy>
02   <wsp:ExactlyOne>
03     <wsp:All>
04       <wsse:SecurityToken>
05         <wsse:TokenType>wsse:Kerberosv5TGT</wsse:TokenType>
06       </wsse:SecurityToken>
07     </wsp:All>
08     <wsp:All>
09       <wsse:SecurityToken>
10         <wsse:TokenType>wsse:X509v3</wsse:TokenType>
11       </wsse:SecurityToken>
12     </wsp:All>
13   </wsp:ExactlyOne>
14 </wsp:Policy>
```

Lines 03-07 and Lines 08-12 express the two alternatives in the policy. If the first alternative is selected, only the Kerberos token type is supported; conversely, if the second alternative is selected, only the X509 token type is supported.

4.2 Policy Identification

A policy expression may be assigned a namespace, indicated either using the `@TargetNamespace` attribute of the `wsp:Policy` element or inherited from a containing element (e.g., `wsdl:definitions`, `xs:schema`). If no namespace is specified by a container or by `@TargetNamespace` of the `wsp:Policy` element, the namespace is "".

A policy expression can also itself be a Web resource, hence identifiable by a URI. To allow policy expressions to be embedded in arbitrary containing elements, the `@wsu:Id` attribute may be used to indicate a fragment ID.

The schema outline for these attributes is as follows:

```
<wsp:Policy xml:base="xs:anyURI" ?
            wsu:Id="xs:ID" ?
            TargetNamespace="xs:anyURI" ?
            ... >
    ...
</wsp:Policy>
```

The following describes the Attribute Information Items listed and defined in the schema outline above:

/wsp:Policy/@wsu:Id

The identity of the policy expression as a relative URI. To refer to this expression, an absolute URI is formed by using the identity as a local name relative to the XML base (i.e., *base#@wsu:Id*).

/wsp:Policy/@TargetNamespace

The namespace URI for the policy expression in the event that one is not available in the surrounding context, or if a different target namespace is desired.

The following example illustrates how to associate a policy expression with a URI:

```
<wsp:Policy xml:base="http://fabrikam123.com/policies" wsu:Id="P1" >
  <wsse:SecurityToken>
    <wsse:TokenType>wsse:Kerberosv5TGT</wsse:TokenType>
  </wsse:SecurityToken>
  <wsse:Integrity>
    <wsse:Algorithm Type="wsse:AlgSignature"
                    URI="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
  </wsse:Integrity>
</wsp:Policy>
```

In the above example, the URI for the policy would be "http://fabrikam123.com/policies#P1".

4.3 Compact Policy Expression

To express a policy in a more compact form while still using the XML Infoset, this specification defines three constructs: an attribute to decorate an assertion, semantics for recursively nested policy operators, and a policy reference/inclusion mechanism. Each is described in the subsections below.

To interpret a compact policy expression in an interoperable form, a compact expression may be converted to the corresponding normal form expression by the following procedure:

1. Start with the **[document element]** property D of the Document Information Item of the policy expression. The **[namespace name]** of D is always "http://schemas.xmlsoap.org/ws/2004/09/policy". In the base case, the **[local name]** property of D is "Policy"; in the recursive case, the **[local name]** property of D is "Policy", "ExactlyOne", or "All".
2. Expand Element Information Items in the **[children]** property of D that are policy references per Section 4.3.3.
3. Convert each Element Information Item C in the **[children]** property of D into normal form.
    3. If the **[namespace name]** property of C is "http://schemas.xmlsoap.org/ws/2004/09/policy" and the **[local name]** property of C is "Policy", "ExactlyOne", or "All", C is an expression of a policy operator; normalize C by recursively applying this procedure.
    4. Otherwise the Element Information Item C is an assertion; normalize C per Section 4.3.1.
4. Apply the policy operator D to the normalized Element Information Items in its **[children]** property and construct a normal form per Section 4.3.2.

Note that an implementation may use a more efficient procedure and is not required to explicitly convert a compact expression into the normal form as long as the processing results are indistinguishable from doing so.

4.3.1 @wsp:Optional

To indicate that a policy assertion is optional, this specification defines an attribute that is a syntactic shortcut for expressing policy alternatives with and without the assertion. The schema outline for this attribute is as follows:

```
<Assertion [ wsp:Optional="xs:boolean" ]? ...> ... </Assertion>
```

The following describes the Attribute Information Item defined in the schema outline above:

/Assertion/@wsp:Optional

If true, the expression of the assertion is semantically equivalent to the following:

```
<wsp:ExactlyOne>
  <wsp:All> <Assertion ...> ... </Assertion> </wsp:All>
  <wsp:All />
</wsp:ExactlyOne>
```

If false, the expression of the assertion is semantically equivalent to the following:

```
<wsp:ExactlyOne>
  <wsp:All> <Assertion ...> ... </Assertion> </wsp:All>
</wsp:ExactlyOne>
```

Omitting this attribute is semantically equivalent to including it with a value of false. Policy expressions should not include this attribute with a value of false, but policy parsers must accept this attribute with a value of false.

For example, the following compact policy expression:

```
01 <wsp:Policy>
```

```
02   <wsse:SecurityToken wsp:Optional="true" >
03     <wsse:TokenType>wsse:Kerberosv5TGT</wsse:TokenType>
04   </wsse:SecurityToken>
05 </wsp:Policy>
```

is equivalent to the following normal form policy expression:

```
A <wsp:Policy>
B   <wsp:ExactlyOne>
C     <wsp:All>
D       <wsse:SecurityToken>
E         <wsse:TokenType>wsse:Kerberosv5TGT</wsse:TokenType>
F       </wsse:SecurityToken>
G     </wsp:All>
H     <wsp:All />
I   </wsp:ExactlyOne>
J </wsp:Policy>
```

The `@wsp:Optional` attribute in Line 2 of the first fragment indicates that the assertion in Lines 02-04 is to be included in a policy alternative whilst excluded from another; it is included in Lines C-G and excluded in Line H. Note that `@wsp:Optional` does not appear in the normal form of a policy expression.

4.3.2 Policy Operators

To compactly express complex policies, policy operators MAY be recursively nested; that is, one or more instances of `wsp:Policy`, `wsp:All`, and/or `wsp:ExactlyOne` MAY be nested within `wsp:Policy`, `wsp:All`, and/or `wsp:ExactlyOne`.

The following rules are used to transform a compact policy expression into a normal form policy expression:

Equivalence

`wsp:Policy` is equivalent to `wsp:All`.

Empty

- `<wsp:All />` expresses a policy with zero policy assertions. Note that since `wsp:Policy` is equivalent to `wsp:All`, `<wsp:Policy />` is therefore equivalent to `<wsp:All />`, i.e., a policy alternative with zero assertions.
- `<wsp:ExactlyOne />` expresses a policy with zero policy alternatives.

Commutative

In line with the previous statements that policy assertions within a policy alternative and policy alternatives within a policy are not ordered (see 3.2 Policy Alternative and 3.3 Policy, respectively), `wsp:All` and `wsp:ExactlyOne` are commutative. For example,

```
<wsp:All> <!-- assertion 1 --> <!-- assertion 2 --> </wsp:All>
```

is equivalent to:

```
<wsp:All> <!-- assertion 2 --> <!-- assertion 1 --> </wsp:All>
```

and:

```
<wsp:ExactlyOne>
  <!-- assertion 1 --> <!-- assertion 2 -->
</wsp:ExactlyOne>
```

is equivalent to:

```
<wsp:ExactlyOne>
  <!-- assertion 2 --> <!-- assertion 1 -->
</wsp:ExactlyOne>
```

Associative

`wsp:All` and `wsp:ExactlyOne` are associative. For example,

```
<wsp:All>
  <!-- assertion 1 -->
  <wsp:All> <!-- assertion 2 --> </wsp:All>
</wsp:All>
```

is equivalent to:

```
<wsp:All> <!-- assertion 1 --> <!-- assertion 2 --> </wsp:All>
```

and:

```
<wsp:ExactlyOne>
  <!-- assertion 1 -->
  <wsp:ExactlyOne> <!-- assertion 2 --> </wsp:ExactlyOne>
</wsp:ExactlyOne>
```

is equivalent to:

```
<wsp:ExactlyOne>
  <!-- assertion 1 --> <!-- assertion 2 -->
</wsp:ExactlyOne>
```

Idempotent

`wsp:All` and `wsp:ExactlyOne` are idempotent. For example,

```
<wsp:All>
  <wsp:All> <!-- assertion 1 --> <!-- assertion 2 --> </wsp:All>
</wsp:All>
```

is equivalent to:

```
<wsp:All> <!-- assertion 1 --> <!-- assertion 2 --> </wsp:All>
```

and:

```
<wsp:ExactlyOne>
  <wsp:ExactlyOne>
    <!-- assertion 1 --> <!-- assertion 2 -->
  </wsp:ExactlyOne>
```

```
</wsp:ExactlyOne>
```

is equivalent to:

```
<wsp:ExactlyOne>
  <!-- assertion 1 --> <!-- assertion 2 -->
</wsp:ExactlyOne>
```

Distributive

`wsp:All` distributes over `wsp:ExactlyOne`. For example,

```
<wsp:All>
  <wsp:ExactlyOne>
    <!-- assertion 1 -->
    <!-- assertion 2 -->
  </wsp:ExactlyOne>
  <wsp:ExactlyOne>
    <!-- assertion 3 -->
    <!-- assertion 4 -->
  </wsp:ExactlyOne>
</wsp:All>
```

is equivalent to:

```
<wsp:ExactlyOne>
  <wsp:All><!-- assertion 1 --><!-- assertion 3 --></wsp:All>
  <wsp:All><!-- assertion 1 --><!-- assertion 4 --></wsp:All>
  <wsp:All><!-- assertion 2 --><!-- assertion 3 --></wsp:All>
  <wsp:All><!-- assertion 2 --><!-- assertion 4 --></wsp:All>
</wsp:ExactlyOne>
```

Similarly,

```
<wsp:All>
  <wsp:ExactlyOne>
    <!-- assertion 1 -->
    <!-- assertion 2 -->
  </wsp:ExactlyOne>
</wsp:All>
```

is equivalent to:

```
<wsp:ExactlyOne>
  <wsp:All>
    <!-- assertion 1 -->
  </wsp:All>
  <wsp:All>
```

```
    <!-- assertion 2 -->
  </wsp:All>
</wsp:ExactlyOne>
```

Distributing `wsp:All` over an empty `wsp:ExactlyOne` is equivalent to no alternatives. For example,

```
<wsp:All>
  <wsp:ExactlyOne>
    <!-- assertion 1 -->
    <!-- assertion 2 -->
  </wsp:ExactlyOne>
  <wsp:ExactlyOne />
</wsp:All>
```

is equivalent to:

```
<wsp:ExactlyOne />
```

Policy operators are intended to express relationships between policy assertions, treating them opaquely. The XML Infoset of an assertion SHOULD NOT contain policy operators in its **[children]** property. To express choice within a domain, assertion authors should consider defining multiple or parameterized assertions. This allows policy authors to express choices using policy alternatives.

For example, given the following compact policy expression:

```
01 <wsp:Policy>
02   <wsp:ExactlyOne>
03     <wsse:SecurityToken>
04       <wsse:TokenType>wsse:Kerberosv5TGT</wsse:TokenType>
05     </wsse:SecurityToken>
06     <wsse:SecurityToken>
07       <wsse:TokenType>wsse:X509v3</wsse:TokenType>
08     </wsse:SecurityToken>
09   </wsp:ExactlyOne>
10   <wssx:Audit wsp:Optional="true" />
11 </wsp:Policy>
```

Applying Section 4.3.1 to `@wsp:Optional` in Line 10, and applying Section 4.3.1 to the implied value of `@wsp:Optional` for the assertions in Lines 03-05 and 06-08 yields:

```
A <wsp:Policy>
B   <wsp:ExactlyOne>
C     <wsp:All>
D       <wsse:SecurityToken>
E         <wsse:TokenType>wsse:Kerberosv5TGT</wsse:TokenType>
```

```
F       </wsse:SecurityToken>
G     </wsp:All>
H     <wsp:All>
I       <wsse:SecurityToken>
J         <wsse:TokenType>wsse:X509v3</wsse:TokenType>
K       </wsse:SecurityToken>
L     </wsp:All>
M   </wsp:ExactlyOne>
N   <wsp:ExactlyOne>
O     <wsp:All><wssx:Audit /></wsp:All>
P     <wsp:All />
Q   </wsp:ExactlyOne>
R </wsp:Policy>
```

Note that the assertion listed in Line 10 expands into the two alternatives in Lines O and P.

Finally, noting that `wsp:Policy` is equivalent to `wsp:All`, and distributing `wsp:All` over `wsp:ExactlyOne` yields the following normal form policy expression:

```
01 <wsp:Policy>
02   <wsp:ExactlyOne>
03     <wsp:All>
04       <wsse:SecurityToken>
05         <wsse:TokenType>wsse:Kerberosv5TGT</wsse:TokenType>
06       </wsse:SecurityToken>
07       <wssx:Audit />
08     </wsp:All>
09     <wsp:All>
10       <wsse:SecurityToken>
11         <wsse:TokenType>wsse:Kerberosv5TGT</wsse:TokenType>
12       </wsse:SecurityToken>
13     </wsp:All>
14     <wsp:All>
15       <wsse:SecurityToken>
16         <wsse:TokenType>wsse:X509v3</wsse:TokenType>
17       </wsse:SecurityToken>
18       <wssx:Audit />
19     </wsp:All>
```

```
20     <wsp:All>
21       <wsse:SecurityToken>
22         <wsse:TokenType>wsse:X509v3</wsse:TokenType>
23       </wsse:SecurityToken>
24     </wsp:All>
25   </wsp:ExactlyOne>
26 </wsp:Policy>
```

Note that the two alternatives listed in Lines C-G and H-L are combined with the two alternatives listed in Lines O and P to create four alternatives in the normalized policy, Lines 03-08, 09-13, 14-19, and 20-24.

4.3.3 Policy Inclusion

In order to share assertions across policy expressions, the `wsp:PolicyReference` element MAY be present anywhere a policy assertion is allowed inside a policy expression. This element is used to include the content of one policy expression in another policy expression.

When a `wsp:PolicyReference` element references a `wsp:Policy` element, then the semantics of inclusion are simply to replace the `wsp:PolicyReference` element with a `wsp:All` element whose **[children]** property is the same as the **[children]** property of the referenced `wsp:Policy` element. That is, the contents of the referenced policy *conceptually replace* the `wsp:PolicyReference` element and are wrapped in an `wsp:All` operator. (Note: References that have a digest attribute SHOULD be validated before being included.)

A policy assertion MUST NOT contain a `wsp:PolicyReference`. If a policy expression needs to reference other policy expressions, the valid way is to encapsulate the assertion within an operator and use references within the operator that will be expanded into the policy expression referenced.

The schema outline for the `wsp:PolicyReference` element is as follows:

```
<wsp:Policy>
  ...
  <wsp:PolicyReference URI="xs:anyURI"
                       Digest="xs:base64Binary" ?
                       DigestAlgorithm="xs:anyURI" ? />
  ...
</wsp:Policy>
```

The following describes the Attribute and Element Information Items defined in the schema outline above:

/wsp:Policy/.../wsp:PolicyReference

This element references a policy expression that is being included.

/wsp:Policy/.../wsp:PolicyReference/@URI

This attribute references a policy expression by URI. There is no requirement that the URI be resolvable.

/wsp:Policy/.../wsp:PolicyReference/@Digest

This optional attribute specifies the digest of the referenced policy expression. This is used to ensure the included policy is the expected policy.

/wsp:Policy/.../wsp:PolicyReference/@DigestAlgorithm

This optional URI attribute specifies the digest algorithms being used. This specification predefines the algorithm below, although additional algorithms can be expressed.

URI	Description
http://schemas.xmlsoap.org/ws/2004/09/policy/Sha1Exc (implied)	The digest is a SHA1 hash over the octet stream resulting from using the Exclusive XML canonicalization defined for XML Signature [XMLSignature].

In the example below two policies include and extend a common policy. In the first example there is a single policy document containing three policy expressions. The first expression is given an identifier but not a fully qualified location. The second and third expressions reference the first element by URI indicating the referenced element is within the document.

```
<wsp:Policy wsu:Id="AUDIT" >
  <wssx:Audit wsp:Optional="true" />
</wsp:Policy>
```

```
<wsp:Policy >
  <wsp:PolicyReference URI="#AUDIT" />
  <wsse:SecurityToken>
    <wsse:TokenType>wsse:X509v3</wsse:TokenType>
  </wsse:SecurityToken>
</wsp:Policy>
```

```
<wsp:Policy >
  <wsp:PolicyReference URI="#AUDIT" />
  <wsse:SecurityToken>
    <wsse:TokenType>wsse:Kerberosv5TGT</wsse:TokenType>
  </wsse:SecurityToken>
</wsp:Policy>
```

There are times when it is desirable to "re-use" a portion of a policy expression. Generally, this can be accomplished by placing the common assertions in a separate policy expression and referencing it.

4.4 Policy Intersection

Policy intersection is useful when two or more parties express policy and want to limit the policy alternatives to those that are mutually compatible. For example, when a requester and a provider express requirements on a message exchange, intersection identifies compatible policy alternatives (if any) included in both requester and provider policies. Intersection is a commutative, associative function that takes two policies and returns a policy.

Because the set of behaviors indicated by a policy alternative depends on the domain-specific semantics of the collected assertion instances, determining whether two policy alternatives are compatible generally involves domain-specific processing. As a first approximation, an algorithm is defined herein that approximates compatibility in a domain-independent manner; specifically, for two policy alternatives to be compatible, they must at least have the same vocabulary (see Section 3.2 Policy Alternative).

Using this approximation, the intersection between two input policies P1 and P2 is a policy Pi consisting of every policy alternative Ai with the following characteristics:

- Given some policy alternative Ax in P1 and some alternative Ay in P2 such that the vocabulary of Ax == the vocabulary of Ay,
- The collection of assertion instances in Ai is all the assertions instances in Ax and all the assertion instances in Ay.

The above definition implies the following:

- The vocabulary of Ai == the vocabulary of Ax == the vocabulary of Ay.
- When the vocabularies of the two input policies overlap but are different, the vocabulary of the intersection of those policies is a subset of the vocabulary of the input policies.
- If the vocabulary of one policy includes an assertion type that is not in the vocabulary of another policy, then the behavior associated with that assertion type is prohibited in the intersection of those policies.

As an example of intersection, consider two input policies:

```
<wsp:Policy> // Policy P1
  <wsp:ExactlyOne>
    <wsp:All> // Alternative A1
      <wsse:Confidentiality>
        <wsse:Algorithm Type="wsse:AlgEncryption"
                        URI="http://www.w3.org/2001/04/xmlenc#3des-cbc" />
        <MessageParts
            Dialect="http://schemas.xmlsoap.org/2002/12/wsse#part" >
          wsp:Body
        </MessageParts>
      </wsse:Confidentiality>
    </wsp:All>
    <wsp:All> // Alternative A2
      <wsse:Confidentiality>
```

```
        <wsse:Algorithm Type="wsse:AlgEncryption"
                        URI="http://www.w3.org/2001/04/xmlenc#3des-cbc" />
        <MessageParts
            Dialect="http://schemas.xmlsoap.org/2002/12/wsse#part" >
          wsp:Body
        </MessageParts>
      </wsse:Confidentiality>
      <wsse:SecurityHeader MustPrepend="true"
                           MustManifestEncryption="true" />
    </wsp:All>
  </wsp:ExactlyOne>
</wsp:Policy>
```

```
<wsp:Policy> // Policy P2
  <wsp:ExactlyOne>
    <wsp:All> // Alternative A3
      <wsse:Confidentiality>
        <wsse:Algorithm Type="wsse:AlgEncryption"
                        URI="http://www.w3.org/2001/04/xmlenc#3des-cbc" />
        <MessageParts
            Dialect="http://schemas.xmlsoap.org/2002/12/wsse#part" >
          wsp:Body wsp:Header(x:AccountNumber)
        </MessageParts>
      </wsse:Confidentiality>
      <wsse:SecurityHeader MustManifestEncryption="true" />
    </wsp:All>
    <wsp:All> // Alternative A4
      <wsse:Confidentiality>
        <wsse:Algorithm Type="wsse:AlgEncryption"
                        URI="http://www.w3.org/2001/04/xmlenc#3des-cbc" />
        <MessageParts
            Dialect="http://schemas.xmlsoap.org/2002/12/wsse#part" >
          wsp:Body wsp:Header(x:AccountNumber)
        </MessageParts>
      </wsse:Confidentiality>
      <wsse:SecurityHeader MustManifestEncryption="true" />
```

```
      <wsse:MessageAge Age="3600" />
    </wsp:All>
  </wsp:ExactlyOne>
</wsp:Policy>
```

Because there is only one alternative (A2) in policy P1 with the same vocabulary as another alternative (A3) in policy P2, the intersection is a policy with a single alternative that contains all of the assertions in A2 and in A3.

```
<wsp:Policy> // Intersection of P1 and P2
  <wsp:ExactlyOne>
    <wsp:All>
      <wsse:Confidentiality>
        <wsse:Algorithm Type="wsse:AlgEncryption"
                        URI="http://www.w3.org/2001/04/xmlenc#3des-cbc" />
        <MessageParts
            Dialect="http://schemas.xmlsoap.org/2002/12/wsse#part" >
          wsp:Body
        </MessageParts>
      </wsse:Confidentiality>
      <wsse:SecurityHeader MustPrepend="true"
                           MustManifestEncryption="true" />
      <wsse:Confidentiality>
        <wsse:Algorithm Type="wsse:AlgEncryption"
                        URI="http://www.w3.org/2001/04/xmlenc#3des-cbc" />
        <MessageParts
            Dialect="http://schemas.xmlsoap.org/2002/12/wsse#part" >
          wsp:Body wsp:Header(x:AccountNumber)
        </MessageParts>
      </wsse:Confidentiality>
      <wsse:SecurityHeader MustManifestEncryption="true" />
    </wsp:All>
  </wsp:ExactlyOne>
</wsp:Policy>
```

Note that the vocabulary of policy P1 is {wsse:Confidentiality, wsse:SecurityHeader}, the vocabulary of policy P2 is {wsse:Confidentiality, wsse:SecurityHeader, wsse:MessageAge}, whilst the vocabulary of the intersection is the intersection {wsse:Confidentiality, wsse:SecurityHeader}.

Note that there is > 1 instance of the assertion type wsse:Confidentiality; when the behavior associated with wsse:Confidentiality is invoked, the contents of both assertion

instances are used to indicate the correct behavior. Whether these two instances are compatible depends on the domain-specific semantics of the wsse:Confidentiality assertion. To leverage intersection, assertion authors are encouraged to factor assertions such that two instances of the same assertion type are always (or at least typically) compatible.

5. Security Considerations

It is strongly RECOMMENDED that policies and assertions be signed to prevent tampering.

Policies SHOULD NOT be accepted unless they are signed and have an associated security token to specify the signer has the right to "speak for" the scope containing the policy. That is, a relying party shouldn't rely on a policy unless the policy is signed and presented with sufficient credentials to pass the relying parties' acceptance criteria.

It should be noted that the mechanisms described in this document could be secured as part of a SOAP message [SOAP11, SOAP12] using WS-Security [WS-Security] or embedded within other objects using object-specific security mechanisms.

6. Acknowledgements

We would like to thank the following people for their contributions towards this specification: Dimitar Angelov (SAP), Martijn de Boer (SAP), Erik Christensen (Microsoft), Giovanni Della-Libera (Microsoft), Yigal Hoffner (IBM), Brian Hulse (IBM), Andrew Jones (IBM), Todd Karakashian (BEA), Scott Konersmann (Microsoft), Frank Leymann (IBM), Steve Lucco (Microsoft), Al Lee (Microsoft), David Levin (Microsoft), Hiroshi Maruyama (IBM), Steve Millet (Microsoft), Vick Mukherjee (Microsoft), Henrik Frystyk Nielsen (Microsoft), Paul Nolan (IBM), Keith Stobie (Microsoft), Tony Storey (IBM), Sanjiva Weerawarana (IBM), Volker Wiechers (SAP).

7. References

[RFC 2119]
S. Bradner, "Key words for use in RFCs to Indicate Requirement Levels," RFC 2119, March 1997.

[RFC 2396]
T. Berners-Lee, et al, "Uniform Resource Identifiers (URI): Generic Syntax," RFC 2396, August 1998.

[SOAP11]
D. Box, et al, "Simple Object Access Protocol (SOAP) 1.1," 08 May 2000.

[SOAP12]
M. Gudgin, et al, "SOAP Version 1.2 Part 1: Messaging Framework," June 2003.

[UDDIAPI20]
D. Ehnebuske, et al, "UDDI Version 2.04 API," July 2002.

[UDDIDataStructure20]
D. Ehnebuske, et al, "UDDI Version 2.03 Data Structure Reference," July 2002.

[UDDI30]
T. Bellwood, et al, "UDDI Version 3.0," July 2002.

[WS-PolicyAttachment]
D. Box, et al, "Web Services Policy Attachment (WS-PolicyAttachment)," September 2004.

[WS-Security]
A. Nadalin, et al, "Web Services Security: SOAP Message Security 1.0 (WS-Security 2004)," March 2004.

[WS-SecurityPolicy]
G. Della-Libera, et al, "Web Services Security Policy Language (WS-SecurityPolicy)," December 2002.

[WSDL]
E. Christensen, et al, "Web Services Description Language (WSDL) 1.1," March 2001.

[XML-NS]
T. Bray, et al, "Namespaces in XML," January 1999.

[XMLSchema1]
D. Beech, et al, "XML Schema Part 1: Structures," May 2001.

[XMLSignature]
D. Eastlake, et al, "XML-Signature Syntax and Processing," February 2002.